```
RRRRRRRRRRRR       MMM           MMM       SSSSSSSSSSSS
RRRRRRRRRRRR       MMM           MMM       SSSSSSSSSSSS
RRRRRRRRRRRR       MMM           MMM       SSSSSSSSSSSS
RRR        RRR     MMMMM       MMMMM       SSS
RRR        RRR     MMMMM       MMMMM       SSS
RRR        RRR     MMMMM       MMMMM       SSS
RRR        RRR     MMM   MMM   MMM         SSS
RRR        RRR     MMM    MMM  MMM         SSS
RRRRRRRRRRRR       MMM         MMM            SSSSSSSSS
RRRRRRRRRRRR       MMM         MMM            SSSSSSSSS
RRRRRRRRRRRR       MMM         MMM            SSSSSSSSS
RRR   RRR          MMM         MMM                  SSS
RRR   RRR          MMM         MMM                  SSS
RRR   RRR          MMM         MMM                  SSS
RRR      RRR       MMM         MMM                  SSS
RRR      RRR       MMM         MMM                  SSS
RRR      RRR       MMM         MMM                  SSS
RRR        RRR  MMM         MMM       SSSSSSSSSSSS
RRR         RRR MMM         MMM       SSSSSSSSSSSS
RRR         RRR MMM         MMM       SSSSSSSSSSSS
```

```
RRRRRRRR   MM      MM   000000    AAAAAA    CCCCCCCC   CCCCCCCC   EEEEEEEEEE   SSSSSSSS   SSSSSSSS
RRRRRRRR   MM      MM   000000    AAAAAA    CCCCCCCC   CCCCCCCC   EEEEEEEEEE   SSSSSSSS   SSSSSSSS
RR    RR   MMMM  MMMM   00    00  AA    AA  CC         CC         EE           SS         SS
RR    RR   MMMM  MMMM   00    00  AA    AA  CC         CC         EE           SS         SS
RR    RR   MM MM MM MM  00  0000  AA    AA  CC         CC         EE           SS         SS
RR    RR   MM MM MM MM  00  0000  AA    AA  CC         CC         EE           SS         SS
RRRRRRRR   MM      MM   00 00  00 AA    AA  CC         CC         EEEEEEE      SSSSS      SSSSS
RRRRRRRR   MM      MM   00 00  00 AA    AA  CC         CC         EEEEEEE      SSSSS      SSSSS
RR  RR     MM      MM   0000    00 AAAAAAAAAA CC       CC         EE                  SS         SS
RR  RR     MM      MM   0000    00 AAAAAAAAAA CC       CC         EE                  SS         SS
RR    RR   MM      MM   00    00  AA    AA  CC         CC         EE                  SS         SS
RR    RR   MM      MM   00    00  AA    AA  CC         CC         EE                  SS         SS     ....
RR    RR   MM      MM   000000    AA    AA  CCCCCCC    CCCCCCC    EEEEEEEEEE   SSSSSSSS   SSSSSSSS  ....
RR    RR   MM      MM   000000    AA    AA  CCCCCCC    CCCCCCC    EEEEEEEEEE   SSSSSSSS   SSSSSSSS  ....


LL           IIIIII   SSSSSSSS
LL           IIIIII   SSSSSSSS
LL             II     SS
LL             II     SS
LL             II     SS
LL             II       SSSSS
LL             II       SSSSS
LL             II            SS
LL             II            SS
LL             II            SS
LL             II            SS
LLLLLLLLLL   IIIIII   SSSSSSSS
LLLLLLLLLL   IIIIII   SSSSSSSS
```

```
0000      1              $BEGIN  RMOACCESS,001,RM$RMS0,<ACCESS/DEACCESS ROUTINES>
0000      2
0000      3   ;
0000      4   ;*******************************************************************************
0000      5   ;*                                                                             *
0000      6   ;*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                   *
0000      7   ;*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
0000      8   ;*   ALL RIGHTS RESERVED.                                                      *
0000      9   ;*                                                                             *
0000     10   ;*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000     11   ;*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE     *
0000     12   ;*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000     13   ;*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000     14   ;*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000     15   ;*   TRANSFERRED.                                                              *
0000     16   ;*                                                                             *
0000     17   ;*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000     18   ;*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
0000     19   ;*   CORPORATION.                                                              *
0000     20   ;*                                                                             *
0000     21   ;*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000     22   ;*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
0000     23   ;*                                                                             *
0000     24   ;*                                                                             *
0000     25   ;*******************************************************************************
0000     26   ;
```

```
0000    28  ;++
0000    29  ; Facility: rms32
0000    30  ;
0000    31  ; Abstract:
0000    32  ;       this module performs the file access and
0000    33  ;       de-access fcp functions.
0000    34  ;
0000    35  ; Environment:
0000    36  ;       star processor running starlet exec.
0000    37  ;
0000    38  ; Author: L F Laverdure,          creation date: 10-MAR-1977
0000    39  ;
0000    40  ; Modified By:
0000    41  ;
0000    42  ;       V04-001  JWT0196          Jim Teague              14-Sep-1984
0000    43  ;               Restore V3 behavior of ignoring UPI for relative
0000    44  ;               and ISAM files.
0000    45  ;
0000    46  ;       V03-043  RAS0326          Ron Schaefer            23-Jul1984
0000    47  ;               Fix RAS0309 to force GET access to be allowed internally
0000    48  ;               if a valid EXE access is requested.  This makes
0000    49  ;               execute-only command procedures work.
0000    50  ;
0000    51  ;       V03-042  JWT0188          Jim Teague              21-Jul-1984
0000    52  ;               Don't allow $OPEN with sharing on magtapes.  RMS
0000    53  ;               was letting this slip through for 512-byte fixed
0000    54  ;               sequential files.
0000    55  ;
0000    56  ;       V03-041  RAS0309          Ron Schaefer            15-Jun-1984
0000    57  ;               Add support for execute-only images and command files.
0000    58  ;
0000    59  ;       V03-040  JWT0179          Jim Teague              23-Apr-1984
0000    60  ;               Always check for an ATR work area before allocating
0000    61  ;               one.
0000    62  ;
0000    63  ;       V03-039  JWT0175          Jim Teague              12-Apr-1984
0000    64  ;               Finish access mode ATR implementation.
0000    65  ;
0000    66  ;       V03-038  SHZ0005          Stephen H. Zalewski     06-Apr-1984
0000    67  ;               Back out the second part of shz0004.  Two reasons, first,
0000    68  ;               global buffers is a connect time option, no open time option.
0000    69  ;               Second, we would be record locking read only isam files, and
0000    70  ;               we never did before.
0000    71  ;
0000    72  ;       V03-037  JWT0173          Jim Teague              1-Apr-1984
0000    73  ;               Disable access mode ATRs for now.
0000    74  ;
0000    75  ;       V03-036  JWT0172          Jim Teague              28-Mar-1984
0000    76  ;               Keep exec mode byte at end of ATR work area.
0000    77  ;
0000    78  ;       V03-035  SHZ0004          Stephen H. Zalewski,    21-Mar-1984
0000    79  ;               Do not take out a file lock if UPI was specified in the
0000    80  ;               SHR field.
0000    81  ;
0000    82  ;               If file is READ ONLY, and global buffers specified, turn
0000    83  ;               on sharing so that global buffers can be used.  Old behavior
0000    84  ;               was to not use sharing since no locking was necessary, however
```

```
0000    85 ;                        this prevented global buffering from being turned on.
0000    86 ;
0000    87 ;        V03-034  RAS0276          Ron Schaefer          20-Mar-1984
0000    88 ;                 Prevent truncate-on-close (TEF FOP option) from being
0000    89 ;                 honored for relative or indexed files.
0000    90 ;
0000    91 ;        V03-033  JWT0167          Jim Teague            15-Mar-1984
0000    92 ;                 Allow write access with buffer offset as long as
0000    93 ;                 BIO is set.  Also implement access-mode ATRs.
0000    94 ;
0000    95 ;        V03-032  DGB0012          Donald G. Blair       01-Mar-1984
0000    96 ;                 Make changes related to ACP calls as part of the
0000    97 ;                 restructuring necessary to support access mode
0000    98 ;                 protected files.
0000    99 ;
0000   100 ;        V03-031  JWT0158          Jim Teague            27-Feb-1984
0000   101 ;                 Adjustment to ANSI buffer offset stuff.  I had placed
0000   102 ;                 the code to request the ATR$C_BUFFER_OFFSET attribute
0000   103 ;                 in a common path for both $OPEN and $CREATE.  It
0000   104 ;                 should only have been in the $OPEN access path.
0000   105 ;
0000   106 ;        V03-030  SHZ0003          Stephen H. Zalewski,  27-Feb-1984
0000   107 ;                 Do not bump the available local buffer count in routine
0000   108 ;                 RM$SETEBK as the local buffer it was trying to give back
0000   109 ;                 (used for FWA) no longer exists.
0000   110 ;
0000   111 ;        V03-029  SHZ0002          Stephen H. Zalewski,  21-Feb-1984
0000   112 ;                 If user opens file no-sharing, multi-streaming read only,
0000   113 ;                 force locking to occur, otherwise no interlocking occurs,
0000   114 ;                 and stream 2 could try to read from a bucket stream 1 is still
0000   115 ;                 reading into cache.
0000   116 ;
0000   117 ;        V03-028  JWT0150          Jim Teague            01-Feb-1984
0000   118 ;                 Implement ANSI buffer offset.
0000   119 ;
0000   120 ;        V03-027  JWT0148          Jim Teague            15-Dec-1983
0000   121 ;                 Enforce ONLY_RU for $OPENs.
0000   122 ;
0000   123 ;        V03-026  RAS0218          Ron Schaefer          5-Dec-1983
0000   124 ;                 Make node names work as search list elements.
0000   125 ;
0000   126 ;        V03-025  DAS0003          David Solomon         13-Sep-1983
0000   127 ;                 Set RJB$V_OPEN before call to RM$MAPJNL.
0000   128 ;
0000   129 ;        V03-024  KBT0582          Keith B. Thompson     12-Aug-1983
0000   130 ;                 Clean up some fwa constants
0000   131 ;
0000   132 ;        V03-023  DAS0002          David Solomon         20-Jul-1983
0000   133 ;                 IFB$V_RUP moved from IFB$B_JNLFLG to IFB$B_JNLFLG2. Migrate
0000   134 ;                 FAB$B_RCF recovery bits in RM$ACCESS (to catch both opens and
0000   135 ;                 creates).
0000   136 ;
0000   137 ;        V03-022  KPL0012          Peter Lieberwirth     1-Jul-1983
0000   138 ;                 Fix bug introduced in V03-020 that caused the PCB address
0000   139 ;                 to be returned as the status code.
0000   140 ;
0000   141 ;        V03-021  DAS0001          David Solomon         22-Jun-1983
```

```
0000   142 ;            If opening a file for RU recovery, use FIB$V_NOLOCK
0000   143 ;            (open regardless).
0000   144 ;
0000   145 ;    V03-020 KPL0013          Peter Lieberwirth       21-Jun-1983
0000   146 ;            Don't migrate FAB recovery bits unless we're in recovery.
0000   147 ;
0000   148 ;    V03-019 KPL0012          Peter Lieberwirth       17-Jun-1983
0000   149 ;            Delay writing AT mapjnl entry until OPEN/CREATE is
0000   150 ;            complete.
0000   151 ;
0000   152 ;    V03-018 TSK0001          Tamar Krichevsky        12-Jun-1983
0000   153 ;            Fix broken branches to journaling routines.
0000   154 ;
0000   155 ;    V03-017 RAS0148          Ron Schaefer            26-Apr-1983
0000   156 ;            Initial support for extended XABPRO.
0000   157 ;
0000   158 ;    V03-016 LJA0059          Laurie J. Anderson      16-Feb-1983
0000   159 ;            Check for Multi-streaming even if NIL is set in the FAB share
0000   160 ;            field.
0000   161 ;
0000   162 ;    V03-015 KBT0491          Keith B. Thompson        9-Feb-1983
0000   163 ;            Checking for "proper" sharing is now done in rm$init_sfsb
0000   164 ;
0000   165 ;    V03-014 TMK0001          Todd M. Katz            01-Feb-1983
0000   166 ;            Add support for Recovery Unit Journalling and RU ROLLBACK
0000   167 ;            Recovery of ISAM files. Under the following set of conditions
0000   168 ;            set the journalling state bit IFB$V_RU_RLK within IFB$B_JNLFLG:
0000   169 ;
0000   170 ;            1. The file is an ISAM file.
0000   171 ;            2. The file is Recovery Unit Journallable.
0000   172 ;            3. The file has been opened for exclusive access (no sharing).
0000   173 ;
0000   174 ;            Setting of this bit will enable pseudo record locking.
0000   175 ;
0000   176 ;    V03-013 LJA0054          Laurie J. Anderson      12-Jan-1983
0000   177 ;            Fill in SHR field in IFB from Users FAB in rm$creacc_set1
0000   178 ;
0000   179 ;    V03-012 KPL0011          Peter Lieberwirth       17-Jan-1983
0000   180 ;            Migrate FAB bits that indicate file is being opened for
0000   181 ;            recovery into the IFB.
0000   182 ;
0000   183 ;    V03-011 SHZ0001          Stephen H. Zalewski     16-Dec-1982
0000   184 ;            Keep disk-structured hbk and ebk in different places in
0000   185 ;            ifb than we keep the swapped hbk and ebk.
0000   186 ;
0000   187 ;    V03-010 ACG0306          Andrew C. Goldstein,    13-Dec-1982  14:55
0000   188 ;            Remove obsolete file header symbols
0000   189 ;
0000   190 ;    V03-009 KBT0412          Keith B. Thompson       30-Nov-1982
0000   191 ;            Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000   192 ;
0000   193 ;    V03-008 JWH0103          Jeffrey W. Horn         20-Sep-1982
0000   194 ;            Move the journaling set-up to RM$SETEBK.
0000   195 ;
0000   196 ;    V03-007 KBT0335          Keith B. Thompson       10-Sep-1982
0000   197 ;            Remove all SO sharing code
0000   198 ;
```

```
0000    199 ;
0000    200 ;        V03-006 JWH0003          Jeffrey W. Horn          31-Aug-1982
0000    201 ;                Put in support for recovery unit journals.
0000    202 ;
0000    203 ;        V03-005 KBT0198          Keith B. Thompson        23-Aug-1982
0000    204 ;                Reorganize psects
0000    205 ;
0000    206 ;        V03-004 KBT0120          Keith B. Thompson        6-Aug-1982
0000    207 ;                Remove ref. to set_sifb_adr and fix all of the version 3
0000    208 ;                rev. numbers
0000    209 ;
0000    210 ;        V03-003 JWH0002          Jeffrey W. Horn          06-Jul-1982
0000    211 ;                Add in call to RM$RTVJNL to get journal control bits and
0000    212 ;                journal names.
0000    213 ;
0000    214 ;        V03-002 KPL0010          Peter Liebrwirth         25-Jun-1982
0000    215 ;                Complete V02-048 by checking for execute-only access
0000    216 ;                whether or not UFO is set.  Previously, if UFO was not
0000    217 ;                set, the check for execute-only access was skipped.
0000    218 ;
0000    219 ;        V03-001 JWH0001          Jeffrey W. Horn          23-Mar-1982
0000    220 ;                Add in call to RM$ASSJNL to set up journaling on this
0000    221 ;                file.
0000    222 ;
0000    223 ;        V02-050 KEK0018          K. E. Kinnear            3-Feb-1982
0000    224 ;                Replace FWA$C_RNSBUFSIZ with the real total size
0000    225 ;                of the concatenated NAME,TYPE, and VER buffer sizes.
0000    226 ;
0000    227 ;        V02-049 CDS0030          C Saether                20-Dec-1981
0000    228 ;                Allow deferred write for shared files.
0000    229 ;
0000    230 ;        V02-048 KPL0009          Peter Lieberwirth        17-Dec-1981
0000    231 ;                Provide support for execute only command files by having ACP
0000    232 ;                check for execute protection in SUPER mode as well as EXEC
0000    233 ;                and KERNEL.
0000    234 ;
0000    235 ;        V02-047 CDS0029          C Saether                16-Sep-1981
0000    236 ;                Allow BIO, BRO with MSE for rel, isam. (same as pre 040).
0000    237 ;
0000    238 ;        V02-046 CDS0028          C Saether                14-Sep-1981
0000    239 ;                Clear NORECLK before UPI check.
0000    240 ;
0000    241 ;        V02-045 CDS0027          C Saether                6-Sep-1981
0000    242 ;                Init BLB queue header when noreclk is cleared.
0000    243 ;
0000    244 ;        V02-044 CDS0026          C Saether                4-Sep-1981
0000    245 ;                NORECLK now set by fseti - clear if locking.
0000    246 ;
0000    247 ;        V02-043 CDS0025          C Saether                31-Aug-1981
0000    248 ;                Always set noreclk.
0000    249 ;
0000    250 ;        V02-042 CDS0024          C Saether                23-Aug-1981
0000    251 ;                Init queue header and allocate a BLB if sharing.
0000    252 ;                Fix bug so that SFSB is allocated for 512 fix len.
0000    253 ;
0000    254 ;        V02-041 KPL0008          Peter Lieberwirth        15-Jul-1981
0000    255 ;                Allocate an SFSB in all cases, including sequential.
```

```
0000  256 ;       V02-040 KPL0007         Peter Lieberwirth        28-Apr-1981
0000  257 ;               Allocate an SFSB via RM$INIT_SFSB if necessary.
0000  258 ;
0000  259 ;       V02-039 CDS0023         C Saether         24-Feb-81       8:30
0000  260 ;               Check fixed length against RSIZ record attribute (ifb$w_lrl)
0000  261 ;               instead of max rec size (ifb$w_mrs).
0000  262 ;
0000  263 ;       V02-038 CDS0022         C Saether         23-Dec-80      15:10
0000  264 ;               Reverse order of attributes on stack so that rewriting
0000  265 ;               record attributes occurs before protection changes.
0000  266 ;
0000  267 ;       V02-037 REFORMAT        C Saether         30-Jul-80      20:20
0000  268 ;
0000  269
```

```
                          0000    271              .SBTTL  DECLARATIONS
                          0000    272
                          0000    273     ;
                          0000    274     ; Include Files:
                          0000    275     ;
                          0000    276
                          0000    277     ;
                          0000    278     ; Macros:
                          0000    279     ;
                          0000    280
                          0000    281              $ARMDEF
                          0000    282              $ATRDEF
                          0000    283              $BDBDEF
                          0000    284              $DEVDEF
                          0000    285              $FABDEF
                          0000    286              $FCHDEF
                          0000    287              $FIBDEF
                          0000    288              $FWADEF
                          0000    289              $IFBDEF
                          0000    290              $IMPDEF
                          0000    291              $IODEF
                          0000    292              $PCBDEF
                          0000    293              $PSLDEF
                          0000    294              $RJBDEF
                          0000    295              $RMSDEF
                          0000    296              $RUCBDEF
                          0000    297              $XABPRODEF
                          0000    298              $XABRDTDEF
                          0000    299
                          0000    300     ;
                          0000    301     ; Equated Symbols:
                          0000    302     ;
                          0000    303
                 00000020 0000    304              FOP=FAB$L_FOP*8
                          0000    305
                          0000    306     ;
                          0000    307     ; Own Storage:
                          0000    308     ;
                          0000    309
```

```
0000   311              .SBTTL  RMSACCESS - PERFORM FCP ACCESS FUNCTION
0000   312
0000   313     ;++
0000   314     ;
0000   315     ; RMSACCESS - perform file access function
0000   316     ;
0000   317     ; This routine sets up the access control word of the fib
0000   318     ; from the various user specifications, builds the
0000   319     ; attribute list to read in the record attributes and
0000   320     ; statistics block, builds the qio parameter list on
0000   321     ; the stack using the filename descriptor, issues
0000   322     ; the qio to the acp to perform the access,
0000   323     ; and finally initializes the ebk and hbk fields of
0000   324     ; the ifab.
0000   325     ;
0000   326     ; Calling sequence:
0000   327     ;
0000   328     ;       BSBW    RMSACCESS
0000   329     ;
0000   330     ; Input Parameters:
0000   331     ;
0000   332     ;       r11     impure area address
0000   333     ;       r10     fwa address
0000   334     ;       r9      ifab addresss
0000   335     ;       r8      fab address
0000   336     ;
0000   337     ; Implicit Inputs:
0000   338     ;
0000   339     ;       fwa$t_fibbuf (fid & did set as required, remainder zero)
0000   340     ;       ifb$v_wrtacc
0000   341     ;       ifb$b_fac
0000   342     ;       fab$l_fop
0000   343     ;       ifb$l_chnl
0000   344     ;       fwa$l_atrladr
0000   345     ;       fwa$q_name
0000   346     ;
0000   347     ; Output Parameters:
0000   348     ;
0000   349     ;       r0      status code
0000   350     ;       r1-r7,ap destroyed
0000   351     ;
0000   352     ;
0000   353     ; Implicit Outputs:
0000   354     ;
0000   355     ;       ifb$v_accessed set
0000   356     ;       the record attributes area of the ifab is initialized
0000   357     ;       the record string is set (fwa$q_rns) over-writing
0000   358     ;               the filename string
0000   359     ;       ifb$l_ios
0000   360     ;       fab$v_ctg set if file contiguous, else cleared
0000   361     ;       fab$l_stv set to system error code on failure
0000   362     ;
0000   363     ; Completion Codes:
0000   364     ;
0000   365     ;       standard rms including suc, fnf, rer, wer, flk, prv,
0000   366     ;       and acc.
0000   367     ;
```

J 9

RMOACCESS   ACCESS/DEACCESS ROUTINES   16-SEP-1984 00:09:38   VAX/VMS Macro V04-00   Page   9
V04-001   RMSACCESS - PERFORM FCP ACCESS FUNCTION 14-SEP-1984 22:32:30   [RMS.SRC]RMOACCESS.MAR;2   (4)

```
0000   368 ; Side Effects:
0000   369 ;
0000   370 ;        may have switched to running at ast level.
0000   371 ;        all user structures except fab must be reprobed.
0000   372 ;--
0000   373
```

```
                              0000    375 RMSACCESS::
                              0000    376          STSTPT   ACCESS
         3C 69   03    E0     0006    377          BBS      #DEVSV_DIR,IFBSL_PRIM_DEV(R9),RMACC ; branch if files-oriented
         04 6A   19    E0     000A    378          BBS      #FWASV_NODE,(R10),NTACC ; branch if network function
                              000E    379          RMSSUC                          ; show success
                05            0011    380          RSB                             ; return to caller
                              0012    381
                              0012    382 ;++
                              0012    383 ;
                              0012    384 ;   perform network access function
                              0012    385 ;
                              0012    386 ;--
                              0012    387
                              0012    388 NTACC:
         0D 69   3E    E0     0012    389          BBS      #IFBSV_DAP,(R9),10$     ; branch if network file access
   04 A8 40020000  8F    D3  0016    390          BITL     #<<FABSM_KFO>!-         ; disallow kfo and ufo options
                              001E    391                   <FABSM_UFO>!-          ;   if task-to-task (to prevent
                              001E    392                   0>,FABSL_FOP(R8)       ;   '$run node::"task=abc"'
                03    13      001E    393          BEQL     10$                    ; branch if neither bits set
              FFDD'   31      0020    394          BRW      NTSSUP_FOP             ; return to caller with rms$_sup
              FFDA'   30      0023    395 10$:     BSBW     NTSACCESS              ; establish logical link
         03 50        E8      0026    396          BLBS     R0,60$
              01FF    31      0029    397          BRW      ERRACCESS             ; branch on failure
         04 68   26    E1     002C    398 60$:     BBC      #FABSV_SQO+FOP,(R8),20$ ; branch if sqo not specified
                              0030    399          SSB      #IFBSV_SQO,(R9)        ; and save bit in ifab
         06 69   3F    E0     0034    400 20$:     BBS      #IFBSV_NSP,(R9),30$    ; branch if task-to-task oper.
              FFC5'   30      0038    401          BSBW     NTSOPEN               ; open file via remote fal
         07 50        E9      003B    402          BLBC     R0,RET                ; branch on failure
                              003E    403 30$:     SSB      #IFBSV_NORECLK,(R9)    ; say no record locking needed
                              0042    404          RMSSUC                         ; show success
                              0045    405
                05            0045    406 RET:     RSB                            ; return to caller
                              0046    407
                              0046    408 RMACC:
                              0046    409 ;
                              0046    410 ;
                              0046    411 ; Migrate FAB recovery bits to the IFB, (don't do so if this process is
                              0046    412 ; not entitled to do recovery).
                              0046    413 ;
   51 00000000'9F    D0      0046    414          MOVL     @#CTLSGL_PCB,R1        ; get PCB address
                1A    E1      004D    415          BBC      #PCBSV_RECOVER,-       ; skip if not a recovery process
         26 24 A1            004F    416                   PCBSL_STS(R1),30$
         4B A8   95          0052    417          TSTB     FABSB_RCF(R8)          ; any bits set?
         21    13            0055    418          BEQL     30$                    ; if eql no
         00    E1            0057    419          BBC      #FABSV_RU,-            ; branch if not RU recovery
         06 4B A8            0059    420                   FABSB_RCF(R8),10$
                              005C    421          SSB      #IFBSV_RU_RECVR,-      ; translate RU to IFB RU_RECVR
                              005C    422                   IFBSB_RECVRFLGS(R9)
                01    E1      0062    423 10$:     BBC      #FABSV_AI,-            ; branch if not roll forward
         06 4B A8            0064    424                   FABSB_RCF(R8),20$
                              0067    425          SSB      #IFBSV_AI_RECVR,-      ; translate AI to IFB AI_RECVR
                              0067    426                   IFBSB_RECVRFLGS(R9)
                02    E1      006D    427 20$:     BBC      #FABSV_BI,-            ; branch if not roll back
         06 48 A8            006F    428                   FABSB_RCF(R8),30$
                              0072    429          SSB      #IFBSV_BI_RECVR,-      ; translate BI to IFB BI_RECVR
                              0072    430                   IFBSB_RECVRFLGS(R9)
                              0078    431
```

```
                              0078      432  ;
                              0078      433  ; Set up for the access.
                              0078      434  ;
                              0078      435
                 01B8  30     0078      436  30$:    BSBW    RMSCREACC_SET1          ; perform first part of setups
              C7 50  E9       007B      437          BLBC    R0,RET                  ; quit on error
                              007E      438
                              007E      439  ;
                              007E      440  ; put a user-mode ATR on the list first
                              007E      441  ;
              85    01  B0    007E      442          MOVW    #1,(R5)+                ; length of access mode byte
           85    2D  B0       0081      443          MOVW    #ATRSC_ACCESS_MODE,(R5)+ ; access mode attribute
        85   0A A9  9E        0084      444          MOVAB   IFBSB_MODE(R9),(R5)+    ; access mode for ACP to read
                              0088      445
              FF75'  30       0088      446          BSBW    RMSOPEN_XAB             ; go process rms open xabs
              B7 50  E9       008B      447          BLBC    R0,RET                  ; continue on success
                              008E      448
                              008E      449  ;
                              008E      450  ; now an exec-mode ATR
                              008E      451  ;
                              008E      452
              85    01  B0    008E      453          MOVW    #1,(R5)+                ; 1 byte length
           85    2D  B0       0091      454          MOVW    #ATRSC_ACCESS_MODE,(R5)+ ; access mode ATR
     85   58 AA  000001FC 8F  C1 0094   455          ADDL3   #508,FQASL_ATR_WORK(R10),(R5)+ ; 1 byte signifying EXEC mode
                              009D      456
              06 69  1C  E1   009D      457          BBC     #DEVSV_RND,IFBSL_PRIM_DEV(R9),8$ ; branch if not disk
                              00A1      458
           00000000'EF  16    00A1      459          JSB     RMSRTVJNL               ; get journal bits, names
                              00A7      460
                 0275  30     00A7      461  8$:     BSBW    RMSCREACC_SET2          ; finish setups
                              00AA      462
                              00AA      463  ;
                              00AA      464  ; set the qio function code and go access the file
                              00AA      465  ;
                              00AA      466
           50  72 8F  9A      00AA      467          MOVZBL  #IO$_ACCESS!IO$M_ACCESS,R0 ; function code
                              00AE      468
                 FF4F'  30    00AE      469          BSBW    RMSFCPFNC               ; do the access
                              00B1      470
              03 50  E8       00B1      471          BLBS    R0,RMSSETHBK            ; continue on RMSFCPFNC success
                 0174  31     00B4      472          BRW     ERRACCESS               ; branch on failure
```

```
                         00B7   474                .SBTTL  RM$SETHBK
                         00B7   475
                         00B7   476   ;++
                         00B7   477   ;
                         00B7   478   ; RM$SETHBK - entry for "create if" that becomes an open
                         00B7   479   ;
                         00B7   480   ; check the file for contiguous and if so set the ctg bit in fop,
                         00B7   481   ; then pick up highest allocated vbn from the statistics block
                         00B7   482   ; and copy to ifab, overwriting the hi vbn field of
                         00B7   483   ; the record attributes. note that the hi-and lo-order words of this vbn
                         00B7   484   ; are reversed on disk and hence are read in reverse order.
                         00B7   485   ; rearrange to give an understandable longword hi vbn. do same for
                         00B7   486   ; eof vbn.
                         00B7   487   ;
                         00B7   488   ;   entry point for "create if" turned into an open.
                         00B7   489   ;
                         00B7   490   ;   set fop output bits according to file attributes.
                         00B7   491   ;
                         00B7   492   ;--
                         00B7   493
                         00B7   494   RM$SETHBK::
        04   04   EF     00B7   495           EXTZV   #IFB$V_ORG,#IFB$S_ORG,-
        51   50 A9       00BA   496                   IFB$B_RFMORG(R9),R1      ; get org
        23 A9   51   90  00BD   497           MOVB    R1,IFB$B_ORGCASE(R9)     ; into separate ifab byte
   00B00200 8F   CA      00C1   498           BICL2   #<FAB$M_CTG!FAB$M_CBT!FAB$M_RCK!FAB$M_WCK>,-
        04 A8           00C7   499                   FAB$L_FOP(R8)            ; clear fop output bits
             07   E1     00C9   500           BBC     #FCH$V_CONTIG,-
        04 44 AA        00CB   501                   FWA$W_OCHAR(R10),10$     ; branch if file not ctg.
                         00CE   502           SSB     #FAB$V_CTG+FOP,(R8)      ; set the ctg bit
             05   E1     00D2   503   10$:    BBC     #FCH$V_CONTIGB,-
        04 44 AA        00D4   504                   FWA$W_OCHAR(R10),20$     ; branch if not ctg best try
                         00D7   505           SSB     #FAB$V_CBT+FOP,(R8)      ; set ctg best try in fop
             03   E1     00DB   506   20$:    BBC     #FCH$V_READCHECK,-
        04 44 AA        00DD   507                   FWA$W_OCHAR(R10),30$     ; branch if no read checking
                         00E0   508           SSB     #FAB$V_RCK+FOP,(R8)      ; set fop rck bit
             04   E1     00E4   509   30$:    BBC     #FCH$V_WRITCHECK,-
        04 44 AA        00E6   510                   FWA$W_OCHAR(R10),40$     ; branch if no write checking
                         00E9   511           SSB     #FAB$V_WCK+FOP,(R8)      ; set fop wck bit
                         00ED   512   40$:
   54 A9   01AC CA      00ED   513           MOVL    FWA$L_HBK(R10),IFB$L_HBK_DISK(R9)   ; move unswapped hbk to ifb
70 A9   54 A9   10   9C 00F3   514           ROTL    #16,IFB$L_HBK_DISK(R9),IFB$L_HBK(R9) ; swap words of hbk
                         00F9   515
   09 66   10   E1      00F9   516           BBC     #FIB$V_EXECUTE,(R6),50$  ; branch if not execute
             01   E1     00FD   517           BBC     #FIB$V_ALT_GRANTED,-
        04 38 A6        00FF   518                   FIB$L_STATUS(R6),50$     ; branch if no read access
        16 A8   02   88 0102   519           BISB2   #FAB$M_GET,FAB$B_FAC(R8); flag read access also permitted
                         0106   520   50$:
        FEF7'   30       0106   521           BSBW    RM$OPEN_XAB1            ; finish up xab processing
                         0109   522
```

N 9

RMOACCESS   ACCESS/DEACCESS ROUTINES   16-SEP-1984 00:09:38 VAX/VMS Macro V04-00   Page 13
V04-001   RMSSETEBK   14-SEP-1984 22:32:30 [RMS.SRC]RMOACCESS.MAR;2   (7)

```
                          0109      524                    .SBTTL  RMSSETEBK
                          0109      525
                          0109      526    ;++
                          0109      527    ;
                          0109      528    ; RMSSETEBK - check for shared access
                          0109      529    ;
                          0109      530    ;     entry point to swap the words of eof block and set ifab bookeeping bit saying fil
                          0109      531    ;
                          0109      532    ;     set up journaling on the file
                          0109      533    ;
                          0109      534    ;     if this is not a sequential file, the shared ifab processing
                          0109      535    ;     is performed, if needed.
                          0109      536    ;
                          0109      537    ;     inputs:
                          0109      538    ;              r11       impure area address
                          0109      539    ;              r10       fwa address
                          0109      540    ;              r9        ifab address
                          0109      541    ;              r8        fab address
                          0109      542    ;
                          0109      543    ;     outputs:
                          0109      544    ;              r0              - status
                          0109      545    ;              r1-r7, ap       - destroyed
                          0109      546    ;              ifb$v_accessed  - set
                          0109      547    ;              ifb$l_ebk       - filled with swapped ebk words form disk
                          0109      548    ;--
                          0109      549
                          0109      550    RMSSETEBK::
                          0109      551            SSB      #IFB$V_ACCESSED,(R9)                  ; declare file accessed
                          010D      552
                          010D      553    ; Deallocate the ATR work area -- we're through with it now
                          010D      554    ;
                3F   BB   010D      555            PUSHR    #^M<R0,R1,R2,R3,R4,R5>   ; Save regs
        54   58 AA   D0   010F      556            MOVL     FWA$L_ATR_WORK(R10),R4   ; Pass address of scratch page
             FEEA'   30   0113      557            BSBW     RMSRET1PAG               ; Return scratch page
             58 AA   D4   0116      558            CLRL     FWA$L_ATR_WORK(R10)      ; Indicate no work area now
                3F   BA   0119      559            POPR     #^M<R0,R1,R2,R3,R4,R5>   ; Restore regs
                          011B      560
   74 A9   58 A9   10 9C  011B      561            ROTL     #16,IFB$L_EBK_DISK(R9),IFB$L_EBK(R9) ; swap words of ebk
                          0121      562
                          0121      563    ; Make sure user doesn't intend to write access an ANSI
                          0121      564    ;         buffer offset (b. o.) tape unless BIO is set
                          0121      565    ;
                          0121      566    ;
        00A8 C9    B5   0121      567            TSTW     IFB$W_BUFFER_OFFSET(R9)              ; is there a non-0 b. o.?
             13    13   0125      568            BEQLU    5$                                   ; if 0, skip next two tests
     09 69   05    E1   0127      569            BBC      #DEV$V_SQD,IFB$L_PRIM_DEV(R9),4$     ; if not a tape, error
     0B 69   30    E1   012B      570            BBC      #IFB$V_WRTACC,(R9),5$                ; if no write access, we're cool
  06 16 A8   05    E0   012F      571            BBS      #FAB$V_BIO,FAB$B_FAC(R8),5$          ; write access is ok with BIO
                     0134      572    4$:        RMSERR   IFF                                  ;    otherwise no write access
                05   0139      573            RSB                                              ;    so cease and desist
                     013A      574
                     013A      575    ;
                     013A      576    ; set up journaling on the file
                     013A      577    ;
                     013A      578    ;
  00A0 C9    20   8A   013A      579    5$:        BICB2    #IFB$M_NEVER_RU,IFB$B_JNLFLG(R9) ; Ignoring NEVER_RU, is
  00A0 C9    95   013F      580            TSTB     IFB$B_JNLFLG(R9)                           ; any journaling bit set?
```

RMO
Syr

$$
$$R
$$R
$$R
$$R
AR
AT
AT
AT
AT
AT
AT
AT
AT
AT
BK
CH
CH
CL
CT
CT
DE
DE
DE
DE
ER
ER
EX
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA
FA

```
          64   13  0143   581           BEQL    SHRCHK                              ; branch if not
                   0145   582   ;
                   0145   583   ; Enforce RU bit settings, specifically ONLY_RU
                   0145   584   ;
    00A0 C9   03   93  0145   585           BITB    #IFB$M_RU!IFB$M_ONLY_RU,IFB$B_JNLFLG(R9) ; RU bits set?
              1B   13  014A   586           BEQL    20$                             ; If not, go on with jnl stuff
 51  00000000'9F  D0  014C   587           MOVL    @#CTL$GL_RUF,R1                  ; RUF loaded?
              05   13  0153   588           BEQL    10$                             ; No RUF, verify ONLY_RU clear
    0D 11 A1  01   E0  0155   589           BBS     #RUCB$V_ACTIVE,RUCB$B_CTRL(R1),20$ ; In RU? Then go set up
                   015A   590   10$:
              01   93  015A   591           BITB    #IFB$M_ONLY_RU,-                 ; If ONLY_RU clear (RU
    00A0 C9        015C   592                   IFB$B_JNLFLG(R9)                     ;   must be set), and not
              06   13  015F   593           BEQL    20$                             ;   in RU then that's ok
                   0161   594           RMSERR  NRU              ; However, if ONLY_RU set and not in RU: error
              05   0166   595           RSB
                   0167   596   ;
                   0167   597   20$:
    00000000'EF  16  0167   598           JSB     RMSASSJNL                       ; set up journaling
          55 50  E9  016D   599           BLBC    RO,RETURN                       ; get out on error
    00A0 C9   95  0170   600           TSTB    IFB$B_JNLFLG(R9)                ; ASSJNL can clear this
          33   13  0174   601           BEQL    SHRCHK                          ; branch if now clear
                   0176   602   ;
                   0176   603   ;
                   0176   604   ; Turn off AT for this MAPJNL call so the AT info can be filled in
                   0176   605   ; during the operation and flushed later.
                   0176   606   ;
 51  00A4 C9  D0  0176   607           MOVL    IFB$L_RJB(R9),R1                ; get RJB address
    7E   0A A1  B0  017B   608           MOVW    RJB$W_FLAGS(R1),-(SP)           ; save current flags
                   017F   609           CSB     #RJB$V_AT,RJB$W_FLAGS(R1)       ; turn off AT for now
    0A A1   10  A8  0184   610           BISW2   #RJB$M_OPEN,RJB$W_FLAGS(R1)     ; set flag that this is an open
    00000000'EF  16  0188   611           JSB     RMSMAPJNL                       ; write out mapping entries
 51  00A4 C9  D0  018E   612           MOVL    IFB$L_RJB(R9),R1                ; get RJB address again
    0A A1   8E  B0  0193   613           MOVW    (SP)+,RJB$W_FLAGS(R1)           ; restore original flags
          2B 50  E9  0197   614           BLBC    RO,RETURN                       ; get out on error
 09 00A2 C9  02  E1  019A   615           BBC     #IFB$V_RUP,IFB$B_JNLFLG2(R9),SHRCHK ; branch if not in RU
    00000000'EF  16  01A0   616           JSB     RMSMAPJNL_RU                    ; write out RU mapping entry
          1C 50  E9  01A6   617           BLBC    RO,RETURN                       ; get out on error
                   01A9   618   ;
    1D 69   33  E1  01A9   619   SHRCHK: BBC     #IFB$V_NORECLK,(R9),CHKSHR      ; not set, then check sharing
                   01AD   620   ;
                   01AD   621   ;
                   01AD   622   ; If this is a Recovery Unit Journalable ISAM file which is being
                   01AD   623   ; opened for exclusive access then set the state bit IFB$V_RU_RLK to
                   01AD   624   ; enable pseudo record locking.
                   01AD   625   ;
                   01AD   626   ;
    23 A9   02  91  01AD   627   EXIT:   CMPB    #IFB$C_IDX,IFB$B_ORGCASE(R9)    ; return if this is not an
              12   12  01B1   628           BNEQ    RETURN                          ; access of an index file
                   01B3   629   ;
    00A0 C9   01  E1  01B3   630           BBC     #IFB$V_RU,IFB$B_JNLFLG(R9),-    ; return if this ISAM file is
              0C       01B8   631           RETURN                          ; not Recovery Unit journallable
                   01B9   632   ;
    17 A8   1F  93  01B9   633           BITB    #FAB$M_SHRGET!FAB$M_SHRPUT-     ; return if any form of sharing
                   01BD   634                   !FAB$M_SHRDEL!FAB$M_SHRUPD-     ; is enabled (inter-process or
                   01BD   635                   !FAB$M_MSE,FAB$B_SHR(R8)        ; inter-stream) - record locking
              06   12  01BD   636           BNEQ    RETURN                          ; will already be enabled
                   01BF   637
```

```
                        01BF  638              SSB     #IFB$V_RU_RLK,IFB$B_JNLFLG2(R9) ; permit pseudo record locking
                        01C5  639
                  05    01C5  640  RETURN: RSB
                        01C6  641
                        01C6  642
                        01C6  643  SETNORECLK:
      E3 69   33  E3    01C6  644              BBCS    #IFB$V_NORECLK,(R9),EXIT ; set NORECLK & exit (always clear)
                        01CA  645
                        01CA  646  CHKSHR:
                        01CA  647
                        01CA  648  ;
                        01CA  649  ; check whether sharing is required
                        01CA  650  ;
                        01CA  651
  05 17 A8   05  E1     01CA  652              BBC     #FAB$V_NIL,FAB$B_SHR(R8),10$    ; If nil spec'd, check MSE
  F2 17 A8   04  E1     01CF  653              BBC     #FAB$V_MSE,FAB$B_SHR(R8),SETNORECLK   ; No locking required
                        01D4  654
                        01D4  655              ASSUME  FAB$C_SEQ       EQ      0
                        01D4  656
      23 A9   95        01D4  657  10$:        TSTB    IFB$B_ORGCASE(R9)               ; is this sequential org?
            1B  13      01D7  658              BEQL    CHKSEQSHR                       ; special checks for 512 fix len recs.
                        01D9  659
                        01D9  660  SHARE:
          FE24'  30     01D9  661              BSBW    RM$INIT_SFSB                    ; get parent lock for record and
                        01DC  662                                                      ; bucket locks.
      OE 50   E9        01DC  663              BLBC    R0,10$                          ; exit on error.
            5A  DD      01DF  664              PUSHL   R10                             ; Save FWA address.
      5A    59.  DO     01E1  665              MOVL    R9,R10                          ; ALBLB wants ifab in r10.
          FE19'  30     01E4  666              BSBW    RM$ALBLB                        ; allocate a BLB to go with BDB (FWA).
          5A 8ED0       01E7  667              POPL    R10                             ; Restore FWA address.
      CO 50   E8        01EA  668              BLBS    R0,EXIT                         ; finish up
                  05    01ED  669  10$:        RSB
                        01EE  670
                        01EE  671  UPIERR: RMSERR  UPI
                  05    01F3  672              RSB
                        01F4  673
                        01F4  674  CHKSEQSHR:
                        01F4  675
                        01F4  676  ;
                        01F4  677  ; want sharing on sequential file - make a few more checks
                        01F4  678  ;
  CD 17 A8   06  E0     01F4  679              BBS     #FAB$V_UPI,FAB$B_SHR(R8),SETNORECLK    ; Branch if UPI.
      28 69   1C  E1    01F9  680              BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R9),SHRERR   ; Magtape!? No way!
  16 A8   60  8F  93    01FD  681              BITB    #FAB$M_BIO!FAB$M_BRO,FAB$B_FAC(R8)     ; any form of block i/o?
            EA  12      0202  682              BNEQ    UPIERR                          ; UPI must be set for block i/o.
                        0204  683
                        0204  684              ASSUME  FAB$C_SEQ       EQ      0
                        0204  685
      01 50 A9   91     0204  686              CMPB    IFB$B_RFMORG(R9),#FAB$C_FIX ; only for fixed length recs
            1B  12      0208  687              BNEQ    SHRERR                          ; neg sorry
  0200 8F  52 A9  B1    020A  688              CMPW    IFB$W_LRL(R9),#512              ; 512 byte records only
            13  12      0210  689              BNEQ    SHRERR                          ; sorry, can't share
      5E A9   01  90    0212  690              MOVB    #1,IFB$B_BKS(R9)                ; bucket size is one
                        0216  691
                        0216  692              ASSUME  <IFB$C_SEQ + 1> EQ      IFB$C_REL
                        0216  693
      23 A9   96        0216  694              INCB    IFB$B_ORGCASE(R9)               ; presto - now you're relative
```

```
     0080 C9   01  D0  0219   695                  MOVL    #1,IFBSL_DVBN(R9)       ; no prologue for seq file
                       021E   696                  SSB     #IFBSV_SEQFIL,(R9)      ; note this is really seq file
          FFB4     31  0222   697        BRW       SHARE                          ; finish shared open
                       0225   698 SHRERR:
                       0225   699                  RMSERR  SHR                    ; can't do that
                   05  022A   700                  RSB                            ; get back
                       022B   701
                       022B   702 ;++
                       022B   703 ;
                       022B   704 ;   handle access failure
                       022B   705 ;
                       022B   706 ;--
                       022B   707
                       022B   708 ERRACCESS:
                       022B   709                  RMSERR  ACC,R1                 ; default error code
          FDCD'    31  0230   710                  BRW     RMSMAPERR              ; go map error code to rms
                       0233   711                                                 ; and return to caller
```

```
                                0233    713                   .SBTTL  RMSCREACC_SET1
                                0233    714
                                0233    715         ;++
                                0233    716         ;
                                0233    717         ; RMSCREACC_SET1 - access, protection, datacheck options fib setup
                                0233    718         ;
                                0233    719         ;   this subroutine initializes the access control word of the fib from
                                0233    720         ;   the various fop options, sets the retrieval window size, and initializes
                                0233    721         ;   r5 to address at which to build & files attributes list
                                0233    722         ;
                                0233    723         ;   inputs:
                                0233    724         ;       r10     fwa address
                                0233    725         ;       r9      ifab address
                                0233    726         ;       r8      fab address
                                0233    727         ;
                                0233    728         ;   outputs:
                                0233    729         ;       r6      fib address
                                0233    730         ;       r5      address for next entry to be added to attribute's list
                                0233    731         ;       r0      success/fail status
                                0233    732         ;
                                0233    733         ;--
                                0233    734
                                0233    735         RMSCREACC_SET1::
        56   14 BA   9E         0233    736                   MOVAB   @FWASQ_FIB+4(R10),R6    ; get fib address
                                0237    737
                                0237    738         ;
                                0237    739         ; initialize the access control word.  it is zero; set desired bits.
                                0237    740         ;
                                0237    741
                                0237    742                   ASSUME  FIBSL_ACCTL EQ 0
        04 69   30   E1         0237    743                   BBC     #IFBSV_WRTACC,(R9),5$   ; branch if read access only
                                023B    744                   SSB     #FIBSV_WRITE,(R6)       ; set write access bit
                                023F    745
                                023F    746         ;
                                023F    747         ; set sharing as desired and determine whether record locking required.
                                023F    748         ;
                                023F    749         ; record locking will be required if there is any form of sharing (inter
                                023F    750         ; or intra process) and there can be any writers of the file.
                                023F    751         ;
                                023F    752
        50   17 AB   90         023F    753         5$:       MOVB    FABSB_SHR(R8),R0        ; get shr field
        4E A9   50   90         0243    754                   MOVB    R0,IFBSB_SHR(R9)        ; save share field in IFB
        04 50   04   E1         0247    755                   BBC     #FABSV_MSE,R0,10$       ; branch if no multi-streams
                                024B    756                   SSB     #IFBSV_MSE,(R9)         ; set mse bit
        10 50   05   E0         024F    757         10$:      BBS     #IFBSV_NIL,R0,20$       ; branch if no sharing
           50   0D   93         0253    758                   BITB    #FABSM_PUT!FABSM_UPD!FABSM_DEL,R0 ; any form of write sharing?
                 13   12        0256    759                   BNEQ    30$                     ; branch if yes
           66   01   88         0258    760                   BISB2   #FIBSM_NOWRITE,(R6)     ; disallow other writers
                                025B    761                                                   ; at most "get" sharing
        08 69   30   E1         025B    762                   BBC     #IFBSV_WRTACC,(R9),25$  ; branch if not write accessed
        08 50   01   E0         025F    763                   BBS     #FABSV_GET,R0,30$       ; branch if allowing other readers
                                0263    764                                                   ; default write accessor to nil
                                0263    765         20$:      SSB     #FIBSV_NOREAD,(R6)      ; disallow other readers
        1A 50   04   E1         0267    766         25$:      BBC     #FABSV_MSE,R0,35$       ; branch if no multi streams
                                026B    767
                                026B    768         ;
                                026B    769         ; record locking required - unless upi set.  require sharers to specify
```

RMOACCESS   ACCESS/DEACCESS ROUTINES   16-SEP-1984 00:09:38 VAX/VMS Macro V04-00   Page 18
V04-001   RMSCREACC_SET1   14-SEP-1984 22:32:30 [RMS.SRC]RMOACCESS.MAR;2   (8)

F 10

```
                                    026B    770  ;        rms locking.
                                    026B    771  ;
                                    026B    772  30$:     CSB     #IFBSV_NORECLK, (R9)   ; clear no locking flag.
        0098 C9  0098 C9  DE        026F    773          MOVAL   IFBSL_BLBFLNK(R9), IFBSL_BLBFLNK(R9) ; Init BLB queue header.
        009C C9  0098 C9  DE        0276    774          MOVAL   IFBSL_BLBFLNK(R9), IFBSL_BLBBLNK(R9) ; Init BLB queue header.
              04 50  06  E0         027D    775          BBS     #FABSV_UPI,R0,35$      ;
                                    0281    776          SSB     #FIBSV_RMSLOCK,(R6)    ; set fib bit for locking
                                    0285    777
                                    0285    778  ;
                                    0285    779  ;        set deferred write ifab flag as required
                                    0285    780  ;
                                    0285    781
           04 68  25  E1            0285    782  35$:     BBC     #FABSV_DFW+FOP,(R8),40$ ; branch if deferred write not
                                    0289    783                                         ; specified
                                    0289    784          SSB     #IFBSV_DFW,(R9)        ; set deferred write flag
                                    028D    785
                                    028D    786  ;
                                    028D    787  ;        set read checking, write checking, and seq. operations only flags
                                    028D    788  ;
                                    028D    789
           07 68  29  E1            028D    790  40$:     BBC     #FABSV_WCK+FOP,(R8),50$ ; branch if no write-checking
              66  20  88            0291    791          BISB2   #1@FIBSV_WRITECK,(R6)   ; enable write-checking
           44 AA  10  88            0294    792          BISB2   #1@FCHSV_WRITCHECK,FWA@W_UCHAR(R10) ; & give file wck attribute
           08 68  37  E1            0298    793  50$:     BBC     #FABSV_RCK+FOP,(R8),60$ ; branch if no read-checking
                                    029C    794          SSB     #FIBSV_READCK,(R6)      ; enable read-checking
           44 AA  08  88            02A0    795          BISB2   #1@FCHSV_READCHECK,FWA@W_UCHAR(R10) ; & give file rck attribute
           08 68  26  E1            02A4    796  60$:     BBC     #FABSV_SQO+FOP,(R8),70$ ; branch if sgo not specified
                                    02A8    797          SSB     #FIBSV_SEQONLY,(R6)     ; set sequential only bit
                                    02AC    798          SSB     #IFBSV_SQO,(R9)         ; and save bit in ifab
                                    02B0    799
                                    02B0    800  ;
                                    02B0    801  ; if magtape, check and set positioning flags (rwo, pos, nef)
                                    02B0    802  ;
                                    02B0    803
           0E 69  05  E1            02B0    804  70$:     BBC     #DEVSV_SQD,IFBSL_PRIM_DEV(R9),80$ ; branch if not magtape
                                    02B4    805          SSB     #FIBSV_PRSRV_ATR,(R6)  ; read rat bits as stored
                                    02B8    806
                                    02B8    807  ;
                                    02B8    808  ; the rms fop bits for magtape positioning are in the same
                                    02B8    809  ; relative position to each other as the corresponding fib bits
                                    02B8    810  ; and additionally have the same polarity - use an extract
                                    02B8    811  ; and insert field to set them appropriately
                                    02B8    812  ; (note:  the wck bit is imbedded - so it gets set or cleared again)
                                    02B8    813  ;
                                    02B8    814
                                    02B8    815          ASSUME  <FABSV_RWO+1> EQ FABSV_POS
                                    02B8    816          ASSUME  <FABSV_POS+1> EQ FABSV_WCK
                                    02B8    817          ASSUME  <FABSV_WCK+1> EQ FABSV_NEF
                                    02B8    818          ASSUME  <FIBSV_REWIND+1> EQ FIBSV_CURPOS
                                    02B8    819          ASSUME  <FIBSV_CURPOS+1> EQ FIBSV_WRITECK
                                    02B8    820          ASSUME  <FIBSV_WRITECK+1> EQ FIBSV_UPDATE
        50 68  04  27  EF           02B8    821          EXTZV   #FABSV_RWO+FOP,#4,(R8),R0; get the fop bits
        66 04  03  50  F0           02BD    822          INSV    R0,#FIBSV_REWIND,#4,(R6)
                                    02C2    823
                                    02C2    824  ;
                                    02C2    825  ; if this is ufo set fib$v_notrunc unless trn bit set in fac
                                    02C2    826  ;
```

```
        09 68    31  E1  02C2    828 80$:    BBC     #FAB$V_UFO+FOP,(R8),90$  ; branch if not ufo
     04 16 A8    04  E0  02C6    829         BBS     #FAB$V_TRN,FAB$B_FAC(R8),90$ ; branch if trn set
                         02CB    830         SSB     #FIB$V_NOTRUNC,(R6)     ; don't allow truncates
                         02CF    831
                         02CF    832 ;
                         02CF    833 ;  check for execute protection
                         02CF    834 ;
                         02CF    835
     12 16 A8    07  E1  02CF    836 90$:    BBC     #FAB$V_EXE,FAB$B_FAC(R8),100$ ; branch if not execute access
           0A A9     91  02D4    837         CMPB    IFB$B_MODE(R9),-
                   02  02D7    838                  #PSL$C_SUPER            ; super (or exec or kernel) mode?
                0C  1A  02D8    839         BGTRU   100$                    ; branch if not (ignore)
     22 A9    02     88  02DA    840         BISB2   #FAB$M_GET,IFB$B_FAC(R9); flag read access also permitted
                01  DO  02DE    841         SSB     #FIB$V_EXECUTE,(R6)     ; have acp check on execute access
        3C A6          02E2    842         MOVL    #ARM$M_READ,-           ; also ask if read access permitted
                         02E4    843                  FIB$L_ALT_ACCESS(R6)
                         02E6    844
                         02E6    845 ;
                         02E6    846 ; Set override exclusive access if opening a file for RU recovery.
                         02E6    847 ;
                         02E6    848
           00   E1  02E6    849 100$:   BBC     #IFB$V_RU_RECVR,-          ; skip if not RU recovery.
        00A1 C9          02E8    850                  IFB$B_RECVRFLGS(R9),-    ;
           0E          02EB    851                  SETRTV                   ;
     00100000 8F    C8  02EC    852         BISL2   #FIB$M_NOLOCK,-          ; set nolock (access regardless) flag.
                66          02F2    853                  FIB$L_ACCTL(R6)          ;
     00000401 8F    CA  02F3    854         BICL2   #FIB$M_NOREAD!FIB$M_NOWRITE,-
                66          02F9    855                  FIB$L_ACCTL(R6)          ; noread/nowrite must be clear.
                         02FA    856
                         02FA    857 ;
                         02FA    858 ; set the retrieval window size
                         02FA    859 ;
                         02FA    860
     03 A6  1C A8  90  02FA    861 SETRTV: MOVB    FAB$B_RTV(R8),FIB$B_WSIZE(R6)
                         02FF    862
                         02FF    863 ;
                         02FF    864 ; the fib is now set up.
                         02FF    865 ; set the attribute control list address into r5
                         02FF    866 ;
                         02FF    867
        55   58 AA  DO  02FF    868         MOVL    FWA$L_ATR_WORK(R10),R5  ; Do we need one?
                11  12  0303    869         BNEQ    10$                     ; If not, don't ask for one
                         0305    870
           0E  BB  0305    871         PUSHR   #^M<R1,R2,R3>           ; Save regs
        FCF6'  30  0307    872         BSBW    RM$GET1PAG              ; Grab a scratch page
        11 50  E9  030A    873         BLBC    R0,20$                  ; Die if none available
     58 AA  53  DO  030D    874         MOVL    R3,FWA$L_ATR_WORK(R10)  ; Save scratch page address
        55  53  DO  0311    875         MOVL    R3,R5                   ;  and put it in R5
           0E  BA  0314    876         POPR    #^M<R1,R2,R3>           ; Restore regs
     01FC C5  01  DO  0316    877 10$:    MOVL    #PSL$C_EXEC,508(R5)     ; Keep exec mode byte in last lword
        50  01  DO  031B    878         MOVL    #1,R0                   ; set success
           05  031F    879 20$:    RSB
```

```
                                    031F      881                .SBTTL  RM$CREACC_SET2
                                    031F      882
                                    031F      883      ;++
                                    031F      884      ;
                                    031F      885      ; RM$CREACC_SET2 - set up stat block, fall thru to creac_3
                                    031F      886      ;
                                    031F      887      ;   subroutine to finish fcp access & create setups started by rm$creacc_set1
                                    031F      888      ;
                                    031F      889      ;   if this is for an access it puts an entry on the attributes list
                                    031F      890      ;   to cause the statistics block to be read
                                    031F      891      ;
                                    031F      892      ;   it then adds attribute list entries for rms record attributes,
                                    031F      893      ;   user characteristics, and, if device is magtape, block size.
                                    031F      894      ;   it then ends the attributes list and builds p6 thru p2 of the fcp's
                                    031F      895      ;   qio parameter block and returns.
                                    031F      896      ;
                                    031F      897      ;   inputs:
                                    031F      898      ;       r10     fwa address
                                    031F      899      ;       r5      attributes list next entry address
                                    031F      900      ;
                                    031F      901      ;   outputs:
                                    031F      902      ;       p6 thru p2 on stack
                                    031F      903      ;       r0, r5 destroyed
                                    031F      904      ;--
                                    031F      905      ;
                                    031F      906      ;   entry point to finish fcp access setups
                                    031F      907      ;
                                    031F      908      ;--
                                    031F      909      ;
                                    031F      910      RM$CREACC_SET2::
          85    0A    B0          031F      911                MOVW    #FWA$S_STATBLK,(R5)+       ; specify # of bytes wanted
          85    09    B0          0322      912                MOVW    #ATR$C_STATBLK,(R5)+      ; read statistics block
   85     01A8  CA    9E          0325      913                MOVAB   FWA$T_STATBLK(R10),(R5)+  ; address for read
                                    032A      914      ;
                                    032A      915      ;
                                    032A      916      ; If magtape, then inquire about buffer offset -- otherwise proceed to
                                    032A      917      ;   CREACC_3.  Note that this inquiry is not made for $CREATE.
                                    032A      918      ;
                                    032A      919      ;
          69    05    E1          032A      920                BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R9),-
                      0B          032D      921                        RM$CREACC_SET3            ; magtape?
          85    02    B0          032E      922                MOVW    #ATR$S_BUFFER_OFFSET,(R5)+ ; size of b.o. field (2)
          85    30    B0          0331      923                MOVW    #ATR$C_BUFFER_OFFSET,(R5)+ ; buffer offset item code
   B5     00A8  C9    3E          0334      924                MOVAW   IFB$W_BUFFER_OFFSET(R9),(R5)+ ; directly to/from ifab
                                    0339      925      ;++
                                    0339      926      ;
                                    0339      927      ;
                                    0339      928      ; RM$CREACC_SET3 - set up for record attributes and user characteristics
                                    0339      929      ;
                                    0339      930      ;   entry point to finish create function setup without getting a statistics block
                                    0339      931      ;
                                    0339      932      ;       put in entries to cause record attributes and user characteristics
                                    0339      933      ;       to be read/written
                                    0339      934      ;
                                    0339      935      ;--
                                    0339      936      ;
                                    0339      937      RM$CREACC_SET3::
```

```
                    50 8ED0  0339  938          POPL    R0                                          ; save return pc
              85    16   B0  033C  939          MOVW    #<IFBSC_FHAEND-IFBSB_RFMORG>,(R5)+ ; # bytes rec attr to xfer
              85    04   B0  033F  940          MOVW    #ATRSC_RECATTR,(R5)+                        ; get rms record attributes
        85    50 A9   DE  0342  941          MOVAL   IFBSB_RFMORG(R9),(R5)+                      ; xfer attr's directly to/from ifab
              85    04   B0  0346  942          MOVW    #ATRSS_UCHAR,(R5)+                          ; size of user characteristics
              85    03   B0  0349  943          MOVW    #ATRSC_UCHAR,(R5)+                          ; specify read/write of  "
        85    44 AA   3E  034C  944          MOVAW   FWASW_UCHAR(R10),(R5)+                      ; addr to read/write    "
        0A 69 05   E1  0350  945          BBC     #DEVSV_SQD,IFBSL_PRIM_DEV(R9),5$; branch if not magtape
              85    02   B0  0354  946          MOVW    #ATRSS_BLOCKSIZE,(R5)+                      ; specify blocksize size (2)
              85    0B   B0  0357  947          MOVW    #ATRSC_BLOCKSIZE,(R5)+                      ; specify read/write of blksiz
        85    48 A9   DE  035A  948          MOVAL   IFBSL_DEVBUFSIZ(R9),(R5)+                   ; xfer directly to/from ifab
                    65   D4  035E  949  5$:     CLRL    (R5)                                        ; flag end of attribute list
                         0360  950
                         0360  951  ;
                         0360  952  ; start building qio argument list on stack
                         0360  953  ;
                         0360  954
                    00   DD  0360  955          PUSHL   #0                                          ; p6
              58 AA   DD  0362  956          PUSHL   FWASL_ATR_WORK(R10)                         ; p5 = attribute list address
        0188 CA   7F  0365  957  P4_P2:  PUSHAQ  FWASQ_RNS(R10)                              ; p4 = resultant name string descriptor
        0170 CA   DF  0369  958          PUSHAL  FWASQ_NAME(R10)                             ; p3 = address of long word
                         036D  959                                                                  ;      to receive resultant string length
        0000012E 8F   D0  036D  960          MOVL    #FWASS_NAMEBUF+FWASS_TYPEBUF+FWASS_VERBUF,-
              0188 CA   0373  961                  FWASQ_RNS(R10)                              ; length of rns buffer
018C CA   04B6 CA   9E  0376  962          MOVAB   FWAST_NAMEBUF(R10),FWASQ_RNS+4(R10) ; overlay input filename
                         037D  963                                                                  ;      with resultant string
        0170 CA   7F  037D  964          PUSHAQ  FWASQ_NAME(R10)                             ; p2 = filename string
                    60   17  0381  965          JMP     (R0)                                        ; return to caller
                         0383  966
                         0383  967  ;++
                         0383  968  ; RMSFCP_P4_P2 - push p4 thru p2 onto stack
                         0383  969  ;
                         0383  970  ;   entry point to push p4 through p2 onto stack for fcp argument list
                         0383  971  ;   build for the $erase function (delete file)
                         0383  972  ;--
                         0383  973
                         0383  974  RMSFCP_P4_P2::
                    01   BA  0383  975          POPR    #^M<R0>                                     ; save return pc
                    DE   11  0385  976          BRB     P4_P2                                       ; go do it
```

J 10

```
0387   978                    .SBTTL  RMSDEACCESS - PERFORM FCP DEACCESS FUNCTION
0387   979
0387   980  ;++
0387   981  ;
0387   982  ; RMSDEACCESS - perform file deaccess function
0387   983  ;
0387   984  ; This routine builds an attribute list to cause the record
0387   985  ; attributes in the ifab to be rewritten to the file
0387   986  ; header, if the file was write accessed, and
0387   987  ; calls rm$fcpfnc to perform the deaccess.
0387   988  ;
0387   989  ; Calling sequence:
0387   990  ;
0387   991  ;       BSBW    RMSDEACCESS
0387   992  ;
0387   993  ; Input Parameters:
0387   994  ;
0387   995  ;       r11     impure area address
0387   996  ;       r9      ifab address
0387   997  ;       r8      fab address
0387   998  ;
0387   999  ; Implicit Inputs:
0387  1000  ;
0387  1001  ;       ifb$l_chnl
0387  1002  ;
0387  1003  ; outputs:
0387  1004  ;
0387  1005  ;       r0      status code
0387  1006  ;       r1-r6,ap destroyed
0387  1007  ;
0387  1008  ; Implicit Outputs:
0387  1009  ;
0387  1010  ;       ifb$l_ios
0387  1011  ;
0387  1012  ; Completion Codes:
0387  1013  ;
0387  1014  ;       standard rms, in particular, suc, dac, fno.
0387  1015  ;
0387  1016  ; Side Effects:
0387  1017  ;
0387  1018  ;       on return rms may be running at ast level
0387  1019  ;       requiring a reprobe of any user structures except
0387  1020  ;       the fab.
0387  1021  ;--
0387  1022
```

```
                        0387 1024
                        0387 1025  ;++
                        0387 1026  ;
                        0387 1027  ;   xab processing arguments for close
                        0387 1028  ;
                        0387 1029  ;--
                        0387 1030
                        0387 1031  CLS_XAB_ARGS:
        00'14 1E        0387 1032          .BYTE   XAB$C_RDT,XAB$C_RDTLEN,XB$C_CLSRDT ; handle rdt xab
        00'10 13        038A 1033          .BYTE   XAB$C_PRO,XAB$C_PROLEN_V3,XB$C_CLSPRO ; handle pro xab
              00        038D 1034          .BYTE   0
                        038E 1035
                        038E 1036  ;++
                        038E 1037  ;
                        038E 1038  ;   perform network deaccess function
                        038E 1039  ;
                        038E 1040  ;--
                        038E 1041
                        038E 1042          ASSUME  IFB$V_DAP GE 56
                        038E 1043          ASSUME  IFB$V_DAP LE 63
                        038E 1044          ASSUME  IFB$V_NSP GE 56
                        038E 1045          ASSUME  IFB$V_NSP LE 63
        00000007        038E 1046  BKP3    = <56/8>                                ; byte offset to flags byte
        000000C0        038E 1047  NETMASK = <1@<IFB$V_DAP-56>> ! <1@<IFB$V_NSP-56>> ; network access-type flags
                        038E 1048
                        038E 1049  NTDAC:
  0A 69    3D   E5      038E 1050          BBCC    #IFB$V_DAP_OPEN,(R9),10$ ; branch if close not necessary
  06 6B    04   E0      0392 1051          BBS     #IMP$V_IORUNDOWN,(R11),10$ ; branch if i/o rundown in progress
        FC67'  30       0396 1052          BSBW    NT$CLOSE                       ; yes, close it there
        11 50   E9      0399 1053          BLBC    R0,20$                         ; branch on failure
        FC61'  30       039C 1054  10$:    BSBW    NT$DEACCESS                    ; destroy logical link with partner
  07 A9  C0 8F  8A      039F 1055          BICB2   #NETMASK,BKP3(R9)             ; clear network access-type flags
        07 50   E9      03A4 1056          BLBC    R0,30$                         ; branch on failure
        FC56'  30       03A7 1057          BSBW    NT$NWA_FREE                    ; discard nwa
                        03AA 1058          RMSSUC                                 ; show success
              05        03AD 1059  20$:    RSB                                    ; exit to caller
        00A8   31       03AE 1060  30$:    BRW     ERRDAC                         ; branch aid
                        03B1 1061
                        03B1 1062  ;++
                        03B1 1063  ;
                        03B1 1064  ;   entry point for rm$deaccess
                        03B1 1065  ;
                        03B1 1066  ;--
                        03B1 1067
                        03B1 1068  RMSDEACCESS::
                        03B1 1069          $TSTPT  DEACCES
  D3 69    0D   E0      03B7 1070          BBS     #DEV$V_NET,IFB$L_PRIM_DEV(R9),NTDAC ; br if network device
                        03BB 1071          RMSSUC  SUC,R6                         ; indicate success
                        03BE 1072
              00  DD    03BE 1073          PUSHL   #0                            ; signal end of attribute list
  5C  C4 AF  9E         03C0 1074          MOVAB   CLS_XAB_ARGS,AP               ; arg list addr for rm$xab_scan
        FC39'  30       03C4 1075          BSBW    RM$XAB_SCAN                    ; process xab chain
        56 50   D0      03C7 1076          MOVL    R0,R6                         ; save status
                        03CA 1077  ;
                        03CA 1078  ; build attribute list on stack to rewrite record attributes
                        03CA 1079  ;
                        03CA 1080
```

```
                1A 69   30   E1  03CA  1081            BBC     #IFBSV_WRTACC,(R9),10$  ; branch if not write accessed
                06 69   38   E1  03CE  1082            BBC     #IFBSV_SEQFIL,(R9),5$   ; skip next few lines if really rel
                                03D2  1083            ASSUME  <IFBSC_SEQ + 1> EQ IFBSC_REL
                23 A9   97       03D2  1084            DECB    IFBSB_ORGCASE(R9)       ; turn back into sequential file
                5E A9   94       03D5  1085            CLRB    IFBSB_BKS(R9)           ; make sure this clear also
                                03D8  1086  5$:
                50 A9   DF       03D8  1087            PUSHAL  IFBSB_RFMORG(R9)        ; write attributes from ifab
          00040016 8F   DD       03DB  1088            PUSHL   #<ATRSC_RECATTRa16>+<IFBSC_FHAEND-IFBSB_RFMORG>
                                03E1  1089                                            ; length & record attriubtes code
                                03E1  1090
                                03E1  1091  ;
                                03E1  1092  ; put org back into rfmorg byte
                                03E1  1093  ;
                                03E1  1094
     50 A9  04  04  23 A9  FO   03E1  1095            INSV    IFBSB_ORGCASE(R9),#IFBSV_ORG,#IFBSS_ORG,IFBSB_RFMORG(R9)
                                03E8  1096
                                03E8  1097  ;
                                03E8  1098  ; allocate a fib to handle various options
                                03E8  1099  ;
                                03E8  1100
                52  40 8F  9A    03E8  1101  10$:      MOVZBL  #FIBSC_LENGTH,R2        ; set size of fib
                       FC11'  30 03EC  1102            BSBW    RMSGETSPC1             ; allocate fib
                                03EF  1103                                            ; build fib descriptor on stack
                       51  DD   03EF  1104            PUSHL   R1                      ; addr of fib
                7E  40 8F  9A    03F1  1105            MOVZBL  #FIBSC_LENGTH,-(SP)     ; and length of fib
                                03F5  1106
                                03F5  1107  ;
                                03F5  1108  ; handle "tef" option (truncate at end of file) if this is a write-accessed
                                03F5  1109  ; disk file.
                                03F5  1110  ;
                                03F5  1111
                23 A9  95       03F5  1112            TSTB    IFBSB_ORGCASE(R9)       ; check for seq file
                       26  12   03F8  1113            BNEQ    20$                     ; don't do it if not seq
                78 A9  D5       03FA  1114            TSTL    IFBSL_SFSB_PTR(R9)      ; check for shared file
                       21  12   03FD  1115            BNEQ    20$                     ; bypass if shared file
             0C 69  36  EO      03FF  1116            BBS     #IFBSV_TEF,(R9),15$     ; branch if auto extend set flag
             19 68  3C  E1      0403  1117            BBC     #FABSV_TEF+FOP,(R8),20$ ; branch if option not speced
             15 69  1C  E1      0407  1118            BBC     #DEVSV_RND,IFBSL_PRIM_DEV(R9),20$ ; or if not disk
             11 69  30  E1      040B  1119            BBC     #IFBSV_WRTACC,(R9),20$  ; or if not write accessed
                                040F  1120
                                040F  1121            ASSUME  FIBSV_TRUNC     GE      8
                                040F  1122
             17 A1  01  88      040F  1123  15$:      BISB2   #<FIBSM_TRUNC a-8>,FIBSW_EXCTL+1(R1) ; ask for truncate
        1C A1  74 A9  DO        0413  1124            MOVL    IFBSL_EBK(R9),FIBSL_EXVBN(R1) ; truncate at eof block
                5C A9  B5       0418  1125            TSTW    IFBSW_FFB(R9)          ; any bytes used this block?
                       03  13   041B  1126            BEQL    20$                     ; branch if none
                1C A1  D6       041D  1127            INCL    FIBSL_EXVBN(R1)         ; yes - don't truncate block
                                0420  1128
                                0420  1129  ;
                                0420  1130  ; check for magtape rewind
                                0420  1131  ;
                                0420  1132
             07 69  05  E1      0420  1133  20$:      BBC     #DEVSV_SQD,IFBSL_PRIM_DEV(R9),40$ ; branch if not magtape
             03 69  27  E1      0424  1134            BBC     #IFBSV_RWC,(R9),20$     ; branch if not speced
                61  08  88      0428  1135            BISB2   #FIBSM_REWIND,FIBSL_ACCTL(R1) ; cause rewind to happen
                                042B  1136
                                042B  1137  ;
```

```
                                        042B  1138 ;  swap the words of ifb$l_hbk and ifb$l_ebk to match files-11
                                        042B  1139 ;  on-disk structure
                                        042B  1140 ;
                                        042B  1141
54 A9  70 A9   10   9C                   042B  1142 40$:    ROTL    #16,IFB$L_HBK(R9),IFB$L_HBK_DISK(R9)
58 A9  74 A9   10   9C                   0431  1143         ROTL    #16,IFB$L_EBK(R9),IFB$L_EBK_DISK(R9)
                                        0437  1144
                                        0437  1145 ;
                                        0437  1146 ;  do the deaccess qio
                                        0437  1147 ;
                                        0437  1148
       50     34   9A                    0437  1149         MOVZBL  #IO$_DEACCESS,R0      ; deaccess function code
              00   DD                    043A  1150         PUSHL   #0                   ; p6 = 0 for qio
       OC AE      DF                    043C  1151         PUSHAL  12(SP)               ; p5 = address of attribute list
            FBBE'  30                    043F  1152         BSBW    RM$FCPFNC_P4         ; do the deaccess acp function
              14   BA                    0442  1153         POPR    #^M<R2,R45>          ; get fib len & addr
              BE   D5                    0444  1154 50$:    TSTL    (SP)+                ; remove attribute list from stack
              FC   12                    0446  1155         BNEQ    50$                  ;   "
       50     DD                    0448  1156         PUSHL   R0                   ; save status code
            FBB3'  30                    044A  1157         BSBW    RM$RETSPC1           ; deallocate the fib
              01   BA                    044D  1158         POPR    #^M<R0>              ; restore the status code
       07 50      E9                    044F  1159         BLBC    R0,ERRDAC            ; branch if error
       03 56      E8                    0452  1160         BLBS    R6,60$               ; branch if no xab error
       50     56   D0                    0455  1161         MOVL    R6,R0                ; report xab error
                   05                    0458  1162 60$:    RSB
                                        0459  1163
                                        0459  1164 ERRDAC:
                                        0459  1165         RMSERR  DAC,R1               ; default error code
            FB9F'  31                    045F  1166         BRW     RM$MAPERR            ; go handle error
                                        0461  1167
                                        0461  1168         .END
```

N 10

RMOACCESS   ACCESS/DEACCESS ROUTINES   16-SEP-1984 00:09:38   VAX/VMS Macro V04-00   Page 26
Symbol table   14-SEP-1984 22:32:30   [RMS.SRC]RMOACCESS.MAR;2   (12)

Symbol	Value			Symbol	Value
$$.PSECT_EP	= 00000000			FABSV_DFW	= 00000005
$$RMSTEST	= 0000001A			FABSV_EXE	= 00000007
$$RMS_PBUGCHK	= 00000010			FABSV_GET	= 00000001
$$RMS_TBUGCHK	= 00000008			FABSV_MSE	= 00000004
$$RMS_UMODE	= 00000004			FABSV_NEF	= 0000000A
ARMSM_READ	= 00000001			FABSV_NIL	= 00000005
ATRSC_ACCESS_MODE	= 0000002D			FABSV_POS	= 00000008
ATRSC_BLOCKSIZE	= 0000000B			FABSV_RCK	= 00000017
ATRSC_BUFFER_OFFSET	= 00000030			FABSV_RU	= 00000000
ATRSC_RECATTR	= 00000004			FABSV_RWO	= 00000007
ATRSC_STATBLK	= 00000000			FABSV_SQO	= 00000006
ATRSC_UCHAR	= 00000003			FABSV_TEF	= 0000001C
ATRSS_BLOCKSIZE	= 00000002			FABSV_TRN	= 00000004
ATRSS_BUFFER_OFFSET	= 00000002			FABSV_UFO	= 00000011
ATRSS_UCHAR	= 00000004			FABSV_UPI	= 00000006
BKP3	= 00000007			FABSV_WCK	= 00000009
CHKSEQSHR	000001F4 R	01		FCHSV_CONTIG	= 00000007
CHKSHR	000001CA R	01		FCHSV_CONTIGB	= 00000005
CLS_XAB_ARGS	00000387 R	01		FCHSV_READCHECK	= 00000003
CTL$GL_PCB	******** X	01		FCHSV_WRITCHECK	= 00000004
CTL$GL_RUF	******** X	01		FIBSB_WSIZE	= 00000003
DEVSV_DIR	= 00000003			FIBSC_LENGTH	= 00000040
DEVSV_NET	= 0000000D			FIBSL_ACCTL	= 00000000
DEVSV_RND	= 0000001C			FIBSL_ALT_ACCESS	= 0000003C
DEVSV_SQD	= 00000005			FIBSL_EXVBN	= 0000001C
ERRACCESS	0000022B R	01		FIBSL_STATUS	= 00000038
ERRDAC	00000459 R	01		FIBSM_NOLOCK	= 00100000
EXIT	000001AD R	01		FIBSM_NOREAD	= 00000400
FABSB_FAC	= 00000016			FIBSM_NOWRITE	= 00000001
FABSB_RCF	= 0000004B			FIBSM_REWIND	= 00000008
FABSB_RTV	= 0000001C			FIBSM_TRUNC	= 00000100
FABSB_SHR	= 00000017			FIBSV_ALT_GRANTED	= 00000001
FABSC_FIX	= 00000001			FIBSV_CURPOS	= 00000004
FABSC_SEQ	= 00000000			FIBSV_EXECUTE	= 00000010
FABSL_FOP	= 00000004			FIBSV_NOREAD	= 0000000A
FABSM_BIO	= 00000020			FIBSV_NOTRUNC	= 0000000B
FABSM_BRO	= 00000040			FIBSV_PRSRV_ATR	= 00000011
FABSM_CBT	= 00200000			FIBSV_READCK	= 00000009
FABSM_CTG	= 00100000			FIBSV_REWIND	= 00000003
FABSM_DEL	= 00000004			FIBSV_RMSLOCK	= 00000012
FABSM_GET	= 00000002			FIBSV_SEQONLY	= 00000006
FABSM_KFO	= 40000000			FIBSV_TRUNC	= 00000008
FABSM_MSE	= 00000010			FIBSV_UPDATE	= 00000006
FABSM_PUT	= 00000001			FIBSV_WRITE	= 00000008
FABSM_RCK	= 00800000			FIBSV_WRITECK	= 00000005
FABSM_SHRDEL	= 00000004			FIBSW_EXCTL	= 00000016
FABSM_SHRGET	= 00000002			FOP	= 00000020
FABSM_SHRPUT	= 00000001			FWASL_ATR_WORK	= 00000058
FABSM_SHRUPD	= 00000008			FWASL_HBK	= 000001AC
FABSM_UFO	= 00020000			FWASQ_FIB	= 00000010
FABSM_UPD	= 00000008			FWASQ_NAME	= 00000170
FABSM_WCK	= 00000200			FWASQ_RNS	= 00000188
FABSV_AI	= 00000001			FWASS_NAMEBUF	= 00000100
FABSV_BI	= 00000002			FWASS_STATBLK	= 0000000A
FABSV_BIO	= 00000005			FWASS_TYPEBUF	= 00000028
FABSV_CBT	= 00000015			FWASS_VERBUF	= 00000006
FABSV_CTG	= 00000014			FWAST_NAMEBUF	= 000004B6

FWA$T_STATBLK	= 000001A8	NETMASK	= 000000C0		
FWA$V_NODE	= 00000019	NT$ACCESS	********	X	01
FWA$W_UCHAR	= 00000044	NT$CLOSE	********	X	01
IFB$B_BKS	= 0000005E	NT$DEACCESS	********	X	01
IFB$B_FAC	= 00000022	NT$NWA_FREE	********	X	01
IFB$B_JNLFLG	= 000000A0	NT$OPEN	********	X	01
IFB$B_JNLFLG2	= 000000A2	NT$SUP_FOP	********	X	01
IFB$B_MODE	= 0000000A	NTACC	00000012	R	01
IFB$B_ORGCASE	= 00000023	NTDAC	0000038E	R	01
IFB$B_RECVRFLGS	= 000000A1	P4_P2	00000365	R	01
IFB$B_RFMORG	= 00000050	PCB$L_STS	= 00000024		
IFB$B_SHR	= 0000004E	PCB$V_RECOVER	= 0000001A		
IFB$C_FHAEND	= 00000066	PIO$A_TRACE	********	X	01
IFB$C_IDX	= 00000002	PSL$C_EXEC	= 00000001		
IFB$C_REL	= 00000001	PSL$C_SUPER	= 00000002		
IFB$C_SEQ	= 00000000	RET	00000045	R	01
IFB$L_BLBBLNK	= 0000009C	RETURN	000001C5	R	01
IFB$L_BLBFLNK	= 00000098	RJB$M_OPEN	= 00000010		
IFB$L_DEVBUFSIZ	= 00000048	RJB$V_AT	= 00000003		
IFB$L_DVBN	= 000000B0	RJB$W_FLAGS	= 0000000A		
IFB$L_EBK	= 00000074	RM$ACCESS	00000000	RG	01
IFB$L_EBK_DISK	= 00000058	RM$ALBLB	********	X	01
IFB$L_HBK	= 00000070	RM$ASSJNL	********	X	01
IFB$L_HBK_DISK	= 00000054	RM$CREACC_SET1	00000233	RG	01
IFB$L_PRIM_DEV	= 00000000	RM$CREACC_SET2	0000031F	RG	01
IFB$L_RJB	= 000000A4	RM$CREACC_SET3	00000339	RG	01
IFB$L_SFSB_PTR	= 00000078	RM$DEACCESS	000003B1	RG	01
IFB$M_NEVER_RU	= 00000020	RM$FCPFNC	********	X	01
IFB$M_ONLY_RU	= 00000001	RM$FCPFNC_P4	********	X	01
IFB$M_RU	= 00000002	RM$FCP_P4_P2	00000383	RG	01
IFB$S_ORG	= 00000004	RM$GETTPAG	********	X	01
IFB$V_ACCESSED	= 00000025	RM$GETSPC1	********	X	01
IFB$V_AI_RECVR	= 00000001	RM$INIT_SFSB	********	X	01
IFB$V_BI_RECVR	= 00000002	RM$MAPERR	********	X	01
IFB$V_DAP	= 0000003E	RM$MAPJNL	********	X	01
IFB$V_DAP_OPEN	= 0000003D	RM$MAPJNL_RU	********	X	01
IFB$V_DFW	= 0000002C	RM$OPEN_XAB	********	X	01
IFB$V_MSE	= 00000031	RM$OPEN_XAB1	********	X	01
IFB$V_NORECLK	= 00000033	RM$RET1PAG	********	X	01
IFB$V_NSP	= 0000003F	RM$RETSPC1	********	X	01
IFB$V_ORG	= 00000004	RM$RTVJNL	********	X	01
IFB$V_RU	= 00000001	RM$SETEBK	00000109	RG	01
IFB$V_RUP	= 00000002	RM$SETHBK	000000B7	RG	01
IFB$V_RU_RECVR	= 00000000	RM$XAB_SCAN	********	X	01
IFB$V_RU_RLK	= 00000003	RMACC	00000046	R	01
IFB$V_RWC	= 00000027	RMS$_ACC	= 0001C002		
IFB$V_SEQFIL	= 00000038	RMS$_DAC	= 0001C012		
IFB$V_SQO	= 0000002D	RMS$_IFF	= 00018804		
IFB$V_TEF	= 00000036	RMS$_NRU	= 000187FC		
IFB$V_WRTACC	= 00000030	RMS$_SHR	= 000186B4		
IFB$W_BUFFER_OFFSET	= 000000A8	RMS$_UPI	= 000187AC		
IFB$W_FFB	= 0000005C	RUCB$B_CTRL	= 00000011		
IFB$W_LRL	= 00000052	RUCB$V_ACTIVE	= 00000001		
IMP$V_IORUNDOWN	= 00000004	SETNORECLK	000001C6	R	01
IO$M_ACCESS	= 00000040	SETRTV	000002FA	R	01
IO$_ACCESS	= 00000032	SHARE	000001D9	R	01
IO$_DEACCESS	= 00000034	SHRCHK	000001A9	R	01

RMOACCESS   ACCESS/DEACCESS ROUTINES   16-SEP-1984 00:09:38 VAX/VMS Macro V04-00   Page 28
Symbol table   14-SEP-1984 22:32:30 [RMS.SRC]RMOACCESS.MAR;2   (12)

C 11

```
SHRERR                          00000225 R    01
TPT$L_ACCESS                    ******** X    01
TPT$L_DEACCES                   ******** X    01
UPIERR                          000001EE R    01
XAB$C_PRO             = 00000013
XAB$C_PROLEN_V3       = 00000010
XAB$C_RDT             = 0000001E
XAB$C_RDTLEN          = 00000014
XBC$C_CLSPRO                    ******** X    01
XBC$C_CLSRDT                    ******** X    01
```

```
                        +------------------+
                        ! Psect synopsis !
                        +------------------+
```

PSECT name	Allocation		PSECT No.	Attributes											
. ABS .	00000000	(0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
RMSRMS0	00000461	(1121.)	01 (1.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE
ABS	00000000	(0.)	02 (2.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE

```
                    +----------------------------+
                    ! Performance indicators !
                    +----------------------------+
```

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:01.24
Command processing	127	00:00:00.69	00:00:05.33
Pass 1	495	00:00:20.28	00:00:53.79
Symbol table sort	2	00:00:03.04	00:00:05.93
Pass 2	206	00:00:04.55	00:00:09.88
Symbol table output	28	00:00:00.20	00:00:00.30
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	890	00:00:28.84	00:01:16.49

The working set limit was 1800 pages.
116398 bytes (228 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 2087 non-local and 70 local symbols.
1168 source lines were read in Pass 1, producing 17 object records in Pass 2.
34 pages of virtual memory were used to define 33 macros.

```
                    +------------------------------+
                    ! Macro library statistics !
                    +------------------------------+
```

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	15
_255DUA28:[SYS.OBJ]LIB.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	29

2213 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RMOACCESS/OBJ=OBJ$:RMOACCESS MSRC$:RMOACCESS/UPDATE=(ENH$:RMOACCESS)+EXECML$/LIB+LIB$:RMS/LIB

NT0SCNXAB
LIS

RM0CACHE
LIS

NT0RENAME
LIS

NT0OPEN
LIS

RM0BUFMGR
LIS

NT0SEARCH
LIS

NT0PUT
LIS

RM0ACCESS
LIS